GD2S03
Advanced Software Engineering & Programming for Games

Bachelor of Software Engineering(BSE)
Game Development

- Overview
  - Handling Touch Events
  - SKTransition
  - SKAction

Handling Touch Events

- **UIKit** directs the events to the most appropriate responder object.
- Responder Objects are used to handle the events.
- A responder object is any instance of UIResponder class.
- The common subclasses of UIResponder class are UIView, UIViewController, and UIApplication.
- For every type of event, UIKit designates a first responder and sends the event to that object first. The first responder varies based on the type of event.
- **Touch events**
  - The first responder is the view in which the touch occurred.
- **Press events**
  - The first responder is the responder that has focus.
- **Shake-motion events**
  - The first responder is the object that you (or UIKit) designate as the first responder.
- **Remote-control events**
  - The first responder is the object that you (or UIKit) designate as the first responder.
  - Trebuchet MS (Body)The first responder is the object that you (or UIKit) designate as
  - the first responder.

Handling Touch Events

```
import SpriteKit

class GameScene: SKScene {

    private let node = SKSpriteNode()

    override func didMove(to view: SKView) {
        createNode()
    }

    func createNode(){
        node.size = CGSize(width: 32, height: 32)
        node.color = UIColor.blue
        node.position = CGPoint(x: self.frame.width/2, y: self.frame.height/2)
        addChild(node)
    }

    override func touchesBegan(_ touches: Set<UITouch>, with event: UIEvent?) {
        if let location = touches.first?.location(in: self){
            if node.contains(location){
                node.size = CGSize(width: node.size.width * 2, height: node.size.height * 2)
            }
        }
    }

    override func touchesMoved(_ touches: Set<UITouch>, with event: UIEvent?) {
        if let location = touches.first?.location(in: self){
            if node.contains(location){
                node.position = location
            }
        }
    }

    override func touchesEnded(_ touches: Set<UITouch>, with event: UIEvent?) {
        if let location = touches.first?.location(in: self){
            if node.contains(location){
               node.size = CGSize(width: node.size.width * 0.5, height: node.size.height * 0.5)
            }
        }
    }

    override func touchesCancelled(_ touches: Set<UITouch>, with event: UIEvent?) {
        print("touch cancelled")
    }
}
```

SKTransition

- An **SKTransition** object is used to perform an animated transition between a SKScene object already presented by an SKView object and a new incoming scene.
- Different self contained scenes can be created to represent different concepts and then transition between those scenes are necessary for e.g.
  - A loading scene to display while other content is loaded
  - A main menu scene to choose what kind of game the user wants to play
  - A scene to configure the details of the specific kind of game the user chose
  - A scene that provides the gameplay
  - A scene displayed when gameplay ends
- Using a transition provides continuity so that the scene change is not quite so abrupt.
- Typically a transition from one scene to another is based on gameplay or user-input, for e.g. if the user presses a button in the main scene , the scene transitioned to another scene.
- When the transition occurs, the scene property is immediately updated to point to the new scene. Then, the animation occurs. Finally, the strong reference to the old scene is removed.
- To keep the scene around after the transition occurs, app needs to keep its own strong reference to the old scene.

SKTransition

- In *GameViewController* class create a scene "*MainMenu*"

```
import UIKit
import SpriteKit

class GameViewController: UIViewController {

    override func viewDidLoad() {
        super.viewDidLoad()

        if let skView = view as! SKView? {
            let skScene = MainMenu(size: skView.bounds.size)
            skScene.scaleMode = .aspectFill
            skView.presentScene(skScene)
            skView.ignoresSiblingOrder = true
            skView.showsFPS = true
            skView.showsNodeCount = true
        }

    }

}
```

- In ***MainMenu*** scene create two SKSpriteNode with different color
- Transit to another scene (GameScene in this case) with different transition animation when either of the node is pressed.

SKTransition

```
import SpriteKit

class MainMenu: SKScene{
    let left = SKSpriteNode()
    let right = SKSpriteNode()

    override func didMove(to view: SKView) {
        self.backgroundColor = UIColor.gray
        left.color = UIColor.red
        left.size = CGSize(width: 64, height: 64)
        left.position = CGPoint(x: self.frame.width/2 - 64, y: self.frame.height/2)
        addChild(left)

        right.color = UIColor.blue
        right.size = CGSize(width: 64, height: 64)
        right.position = CGPoint(x: self.frame.width/2 + 64, y: self.frame.height/2)
        addChild(right)

    }

    override func touchesBegan(_ touches: Set<UITouch>, with event: UIEvent?) {
        let location = touches.first?.location(in: self)
        if left.contains(location!){
            let newScene = GameScene(size: (self.view?.bounds.size)!)
            let transition = SKTransition.reveal(with: .down, duration: 2)
            self.view?.presentScene(newScene, transition: transition)
            transition.pausesOutgoingScene = true
            transition.pausesIncomingScene = false
        }
        else if right.contains(location!){
            let newScene = GameScene(size: (self.view?.bounds.size)!)
            let transition = SKTransition.crossFade(withDuration: 2)
            self.view?.presentScene(newScene, transition: transition)
            transition.pausesOutgoingScene = true
            transition.pausesIncomingScene = true
        }
    }
}
```

SKTransition

- Transition from one scene to another is performed by the view

```
import SpriteKit

class GameScene: SKScene {

    let square = SKSpriteNode()

    override func didMove(to view: SKView) {
        self.backgroundColor = UIColor.magenta
        square.color = UIColor.yellow
        square.size = CGSize(width: 128, height: 128)
        square.position = CGPoint(x: self.frame.width/2, y: self.frame.height/2)
        addChild(square)
    }

}
```

SKAction

- An SKAction object is an action that is executed by a node in the scene (SKScene).
- Actions are most often used to change the structure and content of the node to which they are attached but can also make other changes to the scene.
- When the scene processes its nodes, actions associated with those nodes are evaluated.

Creating and executing Action

```
import SpriteKit

class GameScene: SKScene {
    let square = SKSpriteNode()

    override func didMove(to view: SKView) {
        square.color = UIColor.red
        square.size = CGSize(width: 64, height: 64)
        square.position = CGPoint(x: self.frame.width/2, y: self.frame.height/2)
        /*
           CREATE AND EXECUTE ACTION HERE
        */
        self.addChild(square)
    }
}
```

SKAction

Movement	`square.run(SKAction.moveBy(x: self.frame.width/2, y: 0, duration: 2))`
Rotate	`square.run(SKAction.rotate(byAngle: CGFloat((Float.pi*45)/180), duration: 2))`
Scale	`square.run(SKAction.scale(by: 4, duration: 2))`
Transperancy	`square.run(SKAction.fadeOut(withDuration: 2))` `square.run(SKAction.fadeAlpha(by: -0.8, duration: 2))`
Repeat	`let action = SKAction.rotate(byAngle: CGFloat((Float.pi * 90) / 180), duration: 1)` `square.run(SKAction.repeatForever(action))`
Check Action	`square.hasActions()`
Action With Key	`square.run(SKAction.scale(by: 4, duration: 4), withKey: "scale")`
Remove Action	`square.removeAllActions()`
Remove Action for key	`square.removeAction(forKey: "scale")`

SKAction

Animate

```
let skAtlas = SKTextureAtlas(named: "bird")
let flyAction = SKAction.animate(with: [skAtlas.textureNamed("1.png"),
                                        skAtlas.textureNamed("2.png"),
                                        skAtlas.textureNamed("3.png"),
                                        skAtlas.textureNamed("4.png"),
                                        skAtlas.textureNamed("5.png"),
                                        skAtlas.textureNamed("6.png"),
                                        skAtlas.textureNamed("7.png"),
                                        skAtlas.textureNamed("8.png")
                                        ],
                                 timePerFrame: 0.1)
square.run(SKAction.repeatForever(flyAction))
```

Action Completion

```
let scaleAction = SKAction.scale(by: 4, duration: 4)
let reverseAction = SKAction.reversed(scaleAction)
square.run(scaleAction) {
    self.square.run(reverseAction())
}
```

SKAction

Sequence	`let scaleAction = SKAction.scale(by: 4, duration: 4)`
	`let rotateAction = SKAction.rotate(byAngle: 3.14, duration: 1)`
	`square.run(SKAction.sequence([scaleAction, rotateAction]))`

Group	`let scaleAction = SKAction.scale(by: 4, duration: 4)`
	`let rotateAction = SKAction.rotate(byAngle: 3.14, duration: 1)`
	`square.run(SKAction.group([scaleAction, rotateAction]))`

Wait	`let fadeOutAction = SKAction.fadeOut(withDuration: 2)`
	`let waitAction = SKAction.wait(forDuration: 2)`
	`let fadeInAction = SKAction.fadeIn(withDuration: 2)`
	`square.run(SKAction.sequence([fadeOutAction, waitAction, fadeInAction]))`

Content	`square.run(SKAction.resize(toWidth: 32, height: 32, duration: 2))` `self.addChild(square)`